POSTER: Hardened Registration Process for Participatory Sensing

Jatesada Borsub
Networked Systems Security Group
KTH Royal Institute of Technology
Stockholm, Sweden
jatesada@kth.se

Panos Papadimitratos
Networked Systems Security Group
KTH Royal Institute of Technology
Stockholm, Sweden
papadim@kth.se

ABSTRACT

Participatory sensing systems need to gather information from a large number of participants. However, the openness of the system is a double-edged sword: by allowing practically any user to join, the system can be abused by an attacker who introduces a large number of virtual devices. This poster proposes a hardened registration process for Participatory Sensing to raise the bar: registrations are screened through a number of defensive measures, towards rejecting spurious registrations that do not correspond to actual devices. This deprives an adversary from a relatively easy take-over and, at the same time, allows a flexible and open registration process. The defensive measures are incorporated in the participatory sensing application.

CCS CONCEPTS

• **Security and privacy → Mobile and wireless security**;

KEYWORDS

Registration Process, Participatory Sensing, International Mobile Equipment Identity (IMEI), Android Rooting, Android Emulator, Mobile phone, CAPTCHA

1 INTRODUCTION

Participatory Sensing (PS) data collection takes advantage of the sensing, processing and storage resources in mobile phones to gain knowledge about the participants and, more so, their environment. The acquired information allows other participants to gain access to shared (contributed) data through a PS service enabled by a mobile application (that operates over wireless cellular or other networks). In this day and age, PS has a strong advantage due to the proliferation of smartphone users and advanced built-in smartphone sensors, which enable large-scale and diverse deployments (e.g. environmental [1] and traffic monitoring [2]).

The security and privacy of PS systems are important, in order to (a) protect the systems and ensure the quality of the data, and (b) safeguard the users' privacy while enabling incentive provision. Solutions have been presented, with architectures such as [3], [4]. However, the user registration process itself can be risky. On the one hand, PS thrives on broad participation; the easier the registration process is, the more likely users are to participate. On the other hand, the openness of the registration process, unless done correctly, can become a vulnerability. An adversary could create a small 'army' of virtual users (devices) and have them registered. Those devices could then be orchestrated to provide fake data and undermine the PS process. One possibility would be to spoof the International Mobile Equipment Identity (IMEI), getting access with many accounts through rooting or using an IMEI spoofing application. As a result, the malicious participants could give fake information, perform a Distributed Denial-of-Service (DDoS) attack, or simply not contribute to the services.

One defensive approach could be a strict registration process - requiring, for example, to hand over the smartphone, corroborate the identity of the user, etc., having stepped into an authorized physical location and bootstrap credentials in place. This may be the only option for some applications, but it would be cumbersome. In this work, we take an alternative approach: we defend against an adversary that may try to register fast a large number of "users" it fully controls, by introducing a number of checks and controls. We integrate those in the PS mobile application, as illustrated in Fig. 1. Although highly skilled adversaries may have ways to defeat such countermeasures, this first line of defence raises the bar and makes it significantly harder for such adversarial behavior to be effective. Furthermore, the proposed solution reduces significantly the burden on the PS service side because checks are to be performed on the user side. This has an added advantage: controls do not force disclosure of sensitive user information to the PS service.

This poster presents our approach for hardened registration process (HRP), designed to protect the PS services from abuse. Notably, among other controls and checks, the design brings forth the idea of a "mobile" CAPTCHA [5] tailored to smartphone platforms.

2 HARDENED REGISTRATION PROCESS

The adversary could mislead the PS system seemingly registering a large set of devices in the area of interest, while, in reality these could be virtual entities, exhibiting arbitrary, malicious behavior. These could be, of course, rooted smartphones, or a smartphone emulator, or a botnet. This way, the adversary could overwhelm the PS service by faking sensor data and smartphone locations. The HRP offers a process (Fig. 1) that can protect PS services from such misbehavior. The HPR checks and controls are complementary and reinforce one another. We describe each briefly next.

WiSec '18, June 18–20, 2018, Stockholm, Sweden

ACM ISBN 978-1-4503-5731-9/18/06…$15.00
https://doi.org/10.1145/3212480.3226109

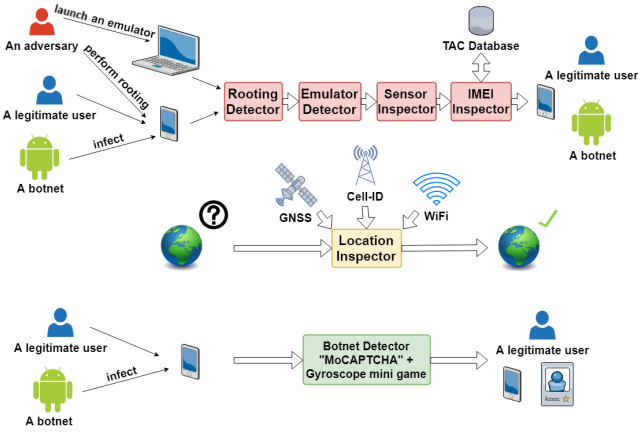

Figure 1: Illustration of the HRP for PS services.

Rooting Detector: The Android rooting process essentially converts Android smartphone normal user credentials and permissions into those of a superuser, allowing full control and free customization of the device. This introduces risks allowing adversaries to manipulate smartphone information that is relevant to the PS service. However, effective techniques [6] can be applied to detect a rooted device through the application program interface (API); they can be incorporated into the HRP, for example, checking directory permissions for writability and checking the existing superuser path.

Emulator Detector: Launching an emulator allows adversaries to manipulate smartphone information and adversely affect PS services. In response, the HRP applies techniques [7] to detect the emulation behavior. Detecting the existence of Bluetooth is a very effective way to do so because Bluetooth is not present in emulators. The support of emulator detection through the Android API and inspection of Android system properties makes the emulator detector more effective.

Sensor Inspector: Sensors in smartphones are important for providing information to PS services, thus, only smartphones with good-condition sensors should be given access to the services. Sensor condition, calibration and diagnostics can leverage the SensorManager API.

IMEI Inspector: An IMEI number can be spoofed, thus, the IMEI inspector compares the smartphone model information that can be retrieved from the Android API against the information from the Type Allocation Code (TAC) database to reject mismatches.

Location Inspector: The HRP allows participants to access PS services if their smartphone is located in a target area. However, the location of a smartphone may be manipulated in order to be eligible by the PS for a given task. Thus, before granting access, the HRP confirms the smartphone location, e.g., combining information from surrounding cell towers, WiFi access points and Global Navigation Satellite Systems [8].

Botnet Detector: The botnet detector attempts to determine whether the apparent smartphone is part a botnet or not; or, inversely, whether it is an actual human being with an actual phone seeking to join the PS system. The HRP offers the smartphone-friendly MoCAPTCHA: it integrates multi-touching, an accelerometer sensor and gravity sensor interaction along with human visual recognition. Our MoCAPTCHA requires the participant to enact gestures that correspond to the provided tasks. An additional step to strengthen the security is introduced with the participant prompted to solve a gyroscope mini game designed in a way that the submitted sensor data cannot be prepared beforehand.

3 IMPLEMENTATION

The HRP application is designed for Android OS version 4.0 or higher. The application works through as a user interface for participants to register their devices and rightfully get access to a PS task. The application exchanges information with a web service [9] which relays information to and from Cell-ID, WiFi access points and TAC databases (implemented with MongoDB). We use common security infiltration tools to test whether malicious behaviors can be detected or not. A virtual smartphone is emulated through various popular Android emulator programs, such as the Android Studios emulator, BlueStacks, GenyMotion, MEnu and Nox. An HTC Desire S is rooted. IMEI spoofing is performed with and without having rooted the device. We compare the results of the tests with those performed with legitimate (untampered) smartphones, such as Samsung GT-I9082L and Samsung A5. The rooting detector, emulator detector, sensor inspector, IMEI inspector and location inspector successfully captured all of the related malicious behaviors. The evaluation of the remaining steps of the HRP are work in progress.

4 CONCLUSION

The openness of the registration process can become a vulnerability for PS services because the adversary could overwhelm the PS services to degrade the quality of the shared/contributed data. The HRP offers a process with defensive measures towards rejecting spurious registrations, to protect PS services from abuse.

ACKNOWLEDGMENTS

This work has been partially supported by the Swedish Foundation for Strategic Research (SSF).

REFERENCES

[1] A. Vasilateanu et al. Environment crowd-sensing for asthma management. *2015 E-Health and Bioengineering Conference (EHB)*, 2015.
[2] P. Zhou et al. Smart Traffic Monitoring with Participatory Sensing. In *Proceedings of the 11th ACM Conference on Embedded Networked Sensor Systems*, SenSys '13, 2013.
[3] S. Gisdakis et al. SPPEAR. In *Proceedings of the 2014 ACM Conference on Security and Privacy in Wireless & Mobile Networks*, WiSec '14, 2014.
[4] S. Gisdakis et al. SHIELD. In *Proceedings of the 8th ACM Conference on Security & Privacy in Wireless and Mobile Networks*, WiSec '15, 2015.
[5] J. Hidalgo and G. Alvarez. CAPTCHAs. *Advances in Computers*, pages 109–181, 2011.
[6] L. Nguyen-Vu et al. Android Rooting: An Arms Race between Evasion and Detection. *Security and Communication Networks*, 2017:1–13, 2017.
[7] T. Vidas and N. Christin. Evading Android Runtime Analysis via Sandbox Detection. In *Proceedings of the 9th ACM Symposium on Information, Computer and Communications Security*, ASIA CCS '14, 2014.
[8] J. Shokouh. Detecting GNSS Attacks on Smartphones. Master's thesis, KTH, School of Electrical Engineering (EES), Communication Networks, 2013.
[9] Accessing data with mongodb: 2018, 2018. https://spring.io/guides/gs/accessing-data-mongodb.